Chapter 1

Introduction

HEC-DSSVue (Hydrologic Engineering Center Data Storage System Visual Utility Engine) is a graphical user interface program for viewing, editing, and manipulating data in HEC-DSS database files. With HEC-DSSVue, you may plot, tabulate, and edit data, as well as manipulate data with over fifty mathematical functions. Along with these functions, HEC-DSSVue provides several utility functions that allow you to enter data sets into a database, rename data set names, copy data sets to other HEC-DSS database files, and delete data sets.

Typically, you will select data sets from a sorted / filtered list of names in a HEC-DSS database file using a mouse. HEC-DSSVue also incorporates the "Jython" standard scripting language, which allows you to specify a routine sequence of steps in a text format, and then execute the sequence from a user-defined button or from a "batch" process.

HEC-DSSVue was written using the Java programming language, which allows it to be run under a variety of different operating systems. Fully supported systems include Microsoft Windows 98 / ME / NT / 2000 / XP, and Sun Solaris (Unix). HEC-DSSVue has been installed on RedHat Linux and other systems. Contact HEC for the availability of these.

1.1 Overview of the HEC Data Storage System

The HEC Data Storage System, or HEC-DSS, is a database system designed to efficiently store and retrieve scientific data that is typically sequential. Such data types include, but are not limited to, time series data, curve data, spatial-oriented gridded data, textual data (such as this manual), and others. The system was designed to make it easy for users and application programs to retrieve and store data.

1.1.1 Background

HEC-DSS was the result of a need that emerged in the late 1970s. During that time most hydrologic studies were performed in a step-wise fashion, passing data from one analysis program to another in a manual mode. While this was functional, it was not very productive. Programs that used the same type of data, or that were sequentially related, did not use a common data format. Also, each program had to have its own set of graphics routines, or other such functions, to aid in the program's use.

The Kissimmee River study, performed by HEC for the Jacksonville District beginning in 1978, required an orderly approach to properly manage the study data and the analysis results. A large number of alternative plans and conditions were to be processed in this project. This study gave birth to the first version of HEC-DSS. The basic design provided for the storage of data in a standard form, independent of any particular program. The data would be provided to the programs when it was needed, and results would be stored in the same independent form for use by utilities and other applications programs. The early design of HEC-DSS was conceived to support files containing many hundreds of data sets, or even as many as a few thousand. As the use of HEC-DSS expanded into real-time data storage applications, data files were written to manage tens and hundreds of thousands of data sets. The current HEC-DSS version is designed for rapid storage and retrieval of data sets from files containing as few as 40 to 50 records to files containing more than 100,000 records.

1.1.2 HEC-DSS Contrasted with Other Database Systems

HEC-DSS is designed to be optimal for storing and retrieving large sets, or series, of data. HEC-DSS incorporates a modified hashing algorithm and hierarchical design for database accesses. This algorithm provides quick access to data sets and an efficient means of adding new data sets to the database. Additionally, HEC-DSS provides a flexible set of utility programs, and it is easy to add to a user's application program. These are the features that distinguish HEC-DSS from most commercial database programs and make it optimal for scientific applications.

HEC-DSS is designed specifically for the storage and retrieval of large sets, or series, of data. This includes daily flow values, hourly precipitation measurements, rating tables, and pages of text information. HEC-DSS is not optimized for dealing with small data sets or single data values, nor is it effective at conditional data searches common to relational database systems. In contrast, most commercial databases are designed for small sets of data, or elements. Such elemental data includes employee records, accounting data, and inventory of stock.

Commercial databases usually employ a relational model in which data is stored in a related manner. A relational database can be viewed essentially as a collection of tables (composed of rows and columns). This type of system requires the construction of a data definition or data dictionary file. Although a relational database requires some initial setup, it can effectively store short data sets comprised of both characters and numbers. Relational database systems use the ANSI ratified Structured Query Language (SQL) to access data.

While relational databases are ideal for elemental data sets, they are not as practical for longer series of data. HEC-DSS, however, is designed for such sets of data. HEC-DSS database files are not defined by a data definition file

like the relational model requires. Thus, there is no set up required by the user. (HEC-DSS data is defined by the pathname and conventions used.) The type of data generally stored in HEC-DSS does not lend itself well to a query language such as SQL (although the selective catalog feature has some similar capabilities).

Also unlike many commercial database systems, the HEC-DSS was designed to be easily added to a user's application program. In traditional "C" and Fortran programs, only two or three function calls are needed to access data. Those calls identify the database, the pathname, and a time window (if desired). Besides these languages, an extensive set of classes are available in both C++ and Java languages (including some access through Visual Basic), the languages with which most new HEC "NexGen" programs are being developed. The NexGen programs make extensive use of HEC-DSS. However, these programs are designed so the interaction with HEC-DSS is seamless; most of the time users will not know that it is being used.

1.2 General Concepts for HEC-DSS

HEC-DSS uses a block of sequential data as the basic unit of storage. This concept results in a more efficient access of time series or other sequentially related data. Each block contains a series of values of a single variable over a time span appropriate for most applications. The basic concept underlying HEC-DSS is the organization of data into records of continuous, applications-related elements, as opposed to individually addressable data items. This approach is more efficient for scientific applications than a conventional database system because it avoids the processing and storage overhead required to assemble an equivalent record from a conventional system.

Data is stored in blocks, or records, within a file, and each record is identified by a unique name called a "pathname." Each time data is stored or retrieved from the file, its pathname must be given. Along with the data, information about the data (*e.g.*, units) is stored in a "header array." HEC-DSS automatically stores the name of the program writing the data, the number of times the data has been written to, and the last written date and time. HEC-DSS documents stored data completely via information contained in the pathname and stored in the header, so no additional information is required to identify it. The self-documenting nature of the database allows information to be recognized and understood months or years after it was stored.

The pathname is the key to the data's location in the database. HEC-DSS analyzes each pathname to determine a "hash" index number. This index determines where the data set is stored within the database. The design ensures that very few disk accesses are made to retrieve or store data sets. One data set is not directly related to another, so there is no need to update other areas of the database when a new data set is stored.

Because of the self-documenting nature of the pathname and the conventions adopted, there is no need for a data dictionary or data definition file as required with other database systems. In fact, there are no database creation tasks or any database setup. Both HEC-DSS utility programs and applications that use HEC-DSS will generate and configure HEC-DSS database files automatically. There is no pre-allocation of space; the software automatically expands the file size as needed.

A HEC-DSS database file has a user-specified conventional name, with an extension of ".dss." As many database files as desired may be generated and there are no size limitations, apart from available disk space. Corps offices have HEC-DSS files that range from a few data sets to many thousands. HEC-DSS adjusts internal tables and hash algorithms to match the database size so as to access both small and very large databases efficiently.

HEC-DSS database files are "direct-access" binary files with no published format. Only programs linked with the HEC-DSS software library can be used to access HEC-DSS files. Direct access files allow efficient retrieval and storage of blocks of data compared to sequential files.

A principal feature of HEC-DSS is that many users can read and write data to a single database at the same time. This multi-user access capability is implemented with system record locking and flushing functions. There is no daemon or other background program managing accesses to a database. A database may exist on a Windows or Unix server machine, and can be accessed by users on PC's or other computers via NFS or the Microsoft network, as long as locking and flushing functions are implemented.

1.2.1 Pathnames

HEC-DSS references data sets, or records, by their *pathnames*. A pathname may consist of up to 391 characters and is, by convention, separated into six parts, which may be up to 64 characters each. Pathnames are automatically translated into all upper case characters. They are separated into six parts (delimited by slashes "/") labeled "A" through "F," as follows:

/A/B/C/D/E/F/

For regular-interval time series data, the part naming convention is:

Part	Description
A	Project, river, or basin name
B	Location
C	Data parameter
D	Starting date of block, in a 9 character military format
E	Time interval
F	Additional user-defined descriptive information

A typical regular-interval time series might be:

/RED RIVER/BEND MARINA/FLOW/01JAN1995/1DAY/OBS/

1.2.2 Catalogs

HEC-DSS utility programs, including HEC-DSSVue, will generate a list of the pathnames in a HEC-DSS file and store that list in a "catalog" file. The catalog file is a list of the record pathnames in the file, along with their last written date and time and the name of the program that wrote that record. The catalog is usually sorted alphabetically by pathname parts. Each pathname has a record tag and a reference number, either of which may be used in place of the pathname in several of the utility programs. The name given to the catalog file is the HEC-DSS file's name with an extension of ".dsc."

A special catalog file, the "condensed catalog," is useful mainly for time series data. In this type of catalog, pathname parts display in columns, and pathnames for time series data, differing only by the date (D part), are referenced with one line.

1.2.3 Data Conventions

HEC-DSS can store data of most any type using a pathname of any structure. To facilitate the ability of application and utility programs to work with and display data, standard record conventions were developed. These conventions define what should be contained in a pathname, how data is stored and what additional information is stored along with the data. For regular-interval time series data (*e.g.*, hourly data), the conventions specify that data is stored in blocks of a standard length, uniform for that time interval, with a pathname that contains the date of the beginning of the block and the time interval. The conventions identify how a pathname for the data should be constructed. Conventions have been defined for regular and irregular interval time series data, paired (curve) data, gridded data (such as NEXRAD radar data) and text (alphanumeric) data.

Regular-interval time series data is data that occurs at a standard time interval. This data is divided into blocks whose length depends on the time interval (for example, hourly data is stored with a block length of a month, while daily data is stored with a block length of a year). Only the date and time of the first piece of data for a block is stored; the times of the other data elements are implied by their location within the block. If a data element, or a set of elements, does not exist for a particular time, a missing data flag is placed in that element's location. Data quality flags may optionally be stored along with a regular-interval time series record.

Irregular-interval time series data does not have a constant time interval between values. This type of data is stored with a date/time stamp for each

element. The user-selectable block size is based on the amount of data that is to be stored. For example, the user may select a block length of a month or a year. Because a date/time stamp is stored with each data element, approximately twice the amount of space is required compared with regular-interval time series data. Data quality flags may optionally be stored along with an irregular-interval time series record.

A convention for paired data has been defined for data that generally defines a curve. It is used for rating tables, flow-frequency curves, and stage-damage curves, etc. One paired data record may contain several curves within it, as long as it has a common set of ordinates. For example a stage-damage curve will contain a set of stages, and it may have associated residential damages, commercial damages, agricultural damages, etc. However, a stage-damage curve and a stage-flow curve cannot be stored in the same record.

Conventions for spatially gridded data can be found in the "GridUtil" User's Manual. Text conventions are specified in the HEC-DSS Programmer's Manual.

1.2.4 Database File Size

HEC-DSS Version 6 database files have been designed for, and tested to, a 2 Gigabyte file size. At the time of the preparation of this manual, modifications were underway to expand Version 6 files to a limit of 16 Gigabytes. These modifications do not modify the file structure.